

Hans Petter Langtangen

Python Scripting for Computational Science

Second Edition

With 62 Figures

 Springer

Table of Contents

1	Introduction	1
1.1	Scripting versus Traditional Programming	1
1.1.1	Why Scripting is Useful in Computational Science	2
1.1.2	Classification of Programming Languages	4
1.1.3	Productive Pairs of Programming Languages	5
1.1.4	Gluing Existing Applications	6
1.1.5	Scripting Yields Shorter Code	7
1.1.6	Efficiency	8
1.1.7	Type-Specification (Declaration) of Variables	9
1.1.8	Flexible Function Interfaces	11
1.1.9	Interactive Computing	12
1.1.10	Creating Code at Run Time	13
1.1.11	Nested Heterogeneous Data Structures	14
1.1.12	GUI Programming	16
1.1.13	Mixed Language Programming	17
1.1.14	When to Choose a Dynamically Typed Language	19
1.1.15	Why Python?	20
1.1.16	Script or Program?	21
1.2	Preparations for Working with This Book	22
2	Getting Started with Python Scripting	27
2.1	A Scientific Hello World Script	27
2.1.1	Executing Python Scripts	28
2.1.2	Dissection of the Scientific Hello World Script	29
2.2	Working with Files and Data	32
2.2.1	Problem Specification	32
2.2.2	The Complete Code	33
2.2.3	Dissection	33
2.2.4	Working with Files in Memory	36
2.2.5	Array Computing	37
2.2.6	Interactive Computing and Debugging	39
2.2.7	Efficiency Measurements	42
2.2.8	Exercises	43
2.3	Gluing Stand-Alone Applications	46
2.3.1	The Simulation Code	47
2.3.2	Using Gnuplot to Visualize Curves	49
2.3.3	Functionality of the Script	50
2.3.4	The Complete Code	51
2.3.5	Dissection	53
2.3.6	Exercises	56
2.4	Conducting Numerical Experiments	58
2.4.1	Wrapping a Loop Around Another Script	59

2.4.2	Generating an HTML Report	60
2.4.3	Making Animations	61
2.4.4	Varying Any Parameter	63
2.5	File Format Conversion	66
2.5.1	A Simple Read/Write Script	66
2.5.2	Storing Data in a Dictionaries and Lists	68
2.5.3	Making a Module with Functions	69
2.5.4	Exercises	71
3	Basic Python	73
3.1	Introductory Topics	74
3.1.1	Recommended Python Documentation	74
3.1.2	Control Statements	75
3.1.3	Running an Application	76
3.1.4	File Reading and Writing	78
3.1.5	Output Formatting	79
3.2	Variables of Different Types	81
3.2.1	Boolean Types	81
3.2.2	The None Variable	81
3.2.3	Numbers and Numerical Expressions	82
3.2.4	Lists and Tuples	84
3.2.5	Dictionaries	90
3.2.6	Splitting and Joining Text	93
3.2.7	String Operations	94
3.2.8	Text Processing	96
3.2.9	The Basics of a Python Class	98
3.2.10	Copy and Assignment	100
3.2.11	Determining a Variable's Type	104
3.2.12	Exercises	105
3.3	Functions	109
3.3.1	Keyword Arguments	110
3.3.2	Doc Strings	111
3.3.3	Variable Number of Arguments	112
3.3.4	Call by Reference	113
3.3.5	Treatment of Input and Output Arguments	115
3.3.6	Function Objects	116
3.4	Working with Files and Directories	117
3.4.1	Listing Files in a Directory	118
3.4.2	Testing File Types	118
3.4.3	Removing Files and Directories	119
3.4.4	Copying and Renaming Files	120
3.4.5	Splitting Pathnames	121
3.4.6	Creating and Moving to Directories	122
3.4.7	Traversing Directory Trees	122
3.4.8	Exercises	125

4	Numerical Computing in Python	131
4.1	A Quick NumPy Primer	133
4.1.1	Creating Arrays	133
4.1.2	Array Indexing	134
4.1.3	Array Computations	136
4.1.4	Type Testing	138
4.1.5	Hidden Temporary Arrays	139
4.1.6	Exercises	140
4.2	Vectorized Algorithms	141
4.2.1	From Scalar to Array in Function Arguments	141
4.2.2	Slicing	143
4.2.3	Remark on Efficiency	144
4.2.4	Exercises	146
4.3	More Advanced Array Computing	147
4.3.1	Random Numbers	147
4.3.2	Linear Algebra	148
4.3.3	The Gnuplot Module	150
4.3.4	Example: Curve Fitting	152
4.3.5	Arrays on Structured Grids	153
4.3.6	File I/O with NumPy Arrays	157
4.3.7	Reading and Writing Tables with NumPy Arrays	158
4.3.8	Functionality in the Numpytools Module	160
4.3.9	Exercises	163
4.4	Other Tools for Numerical Computations	168
4.4.1	The ScientificPython Package	168
4.4.2	The SciPy Package	174
4.4.3	The Python–Matlab Interface	179
4.4.4	Some Useful Python Modules	180
5	Combining Python with Fortran, C, and C++	183
5.1	About Mixed Language Programming	183
5.1.1	Applications of Mixed Language Programming	184
5.1.2	Calling C from Python	184
5.1.3	Automatic Generation of Wrapper Code	186
5.2	Scientific Hello World Examples	188
5.2.1	Combining Python and Fortran	189
5.2.2	Combining Python and C	194
5.2.3	Combining Python and C++ Functions	199
5.2.4	Combining Python and C++ Classes	202
5.2.5	Exercises	205
5.3	A Simple Computational Steering Example	206
5.3.1	Modified Time Loop for Repeated Simulations	207
5.3.2	Creating a Python Interface	208
5.3.3	The Steering Python Script	210
5.3.4	Equipping the Steering Script with a GUI	213
5.4	Scripting Interfaces to Large Libraries	214

6	Introduction to GUI Programming	219
6.1	Scientific Hello World GUI	220
6.1.1	Introductory Topics	220
6.1.2	The First Python/Tkinter Encounter	222
6.1.3	Binding Events	225
6.1.4	Changing the Layout	226
6.1.5	The Final Scientific Hello World GUI	230
6.1.6	An Alternative to Tkinter Variables	232
6.1.7	About the Pack Command	233
6.1.8	An Introduction to the Grid Geometry Manager	235
6.1.9	Implementing a GUI as a Class	237
6.1.10	A Simple Graphical Function Evaluator	239
6.1.11	Exercises	240
6.2	Adding GUIs to Scripts	243
6.2.1	A Simulation and Visualization Script with a GUI	243
6.2.2	Improving the Layout	246
6.2.3	Exercises	248
6.3	A List of Common Widget Operations	249
6.3.1	Frame	252
6.3.2	Label	252
6.3.3	Button	254
6.3.4	Text Entry	254
6.3.5	Balloon Help	256
6.3.6	Option Menu	257
6.3.7	Slider	257
6.3.8	Check Button	258
6.3.9	Making a Simple Megawidget	258
6.3.10	Menu Bar	259
6.3.11	List Data	261
6.3.12	Listbox	262
6.3.13	Radio Button	265
6.3.14	Combo Box	266
6.3.15	Message Box	267
6.3.16	User-Defined Dialogs	269
6.3.17	Color-Picker Dialogs	270
6.3.18	File Selection Dialogs	273
6.3.19	Toplevel	274
6.3.20	Some Other Types of Widgets	275
6.3.21	Adapting Widgets to the User's Resize Actions	276
6.3.22	Customizing Fonts and Colors	278
6.3.23	Widget Overview	280
6.3.24	Exercises	282

7	Web Interfaces and CGI Programming	289
7.1	Introductory CGI Scripts	290
7.1.1	Web Forms and CGI Scripts	291
7.1.2	Generating Forms in CGI Scripts	293
7.1.3	Debugging CGI Scripts	295
7.1.4	A General Shell Script Wrapper for CGI Scripts	296
7.1.5	Security Issues	298
7.2	Adding Web Interfaces to Scripts	300
7.2.1	A Class for Form Parameters	301
7.2.2	Calling Other Programs	303
7.2.3	Running Simulations	304
7.2.4	Getting a CGI Script to Work	305
7.2.5	Using Web Applications from Scripts	308
7.2.6	Exercises	310
8	Advanced Python	313
8.1	Miscellaneous Topics	313
8.1.1	Parsing Command-Line Arguments	313
8.1.2	Platform-Dependent Operations	316
8.1.3	Run-Time Generation of Code	317
8.1.4	Exercises	318
8.2	Regular Expressions and Text Processing	320
8.2.1	Motivation	320
8.2.2	Special Characters	323
8.2.3	Regular Expressions for Real Numbers	325
8.2.4	Using Groups to Extract Parts of a Text	328
8.2.5	Extracting Interval Limits	329
8.2.6	Extracting Multiple Matches	333
8.2.7	Splitting Text	338
8.2.8	Pattern-Matching Modifiers	339
8.2.9	Substitution and Backreferences	341
8.2.10	Example: Swapping Arguments in Function Calls	342
8.2.11	A General Substitution Script	345
8.2.12	Debugging Regular Expressions	347
8.2.13	Exercises	349
8.3	Tools for Handling Data in Files	357
8.3.1	Writing and Reading Python Data Structures	358
8.3.2	Pickling Objects	359
8.3.3	Shelving Objects	361
8.3.4	Writing and Reading Zip and Tar Archive Files	362
8.3.5	Downloading Internet Files	362
8.3.6	Binary Input/Output	364
8.3.7	Exercises	366
8.4	A Database for NumPy Arrays	367
8.4.1	The Structure of the Database	367
8.4.2	Pickling	370
8.4.3	Formatted ASCII Storage	370

8.4.4	Shelving	372
8.4.5	Comparing the Various Techniques	373
8.5	Scripts Involving Local and Remote Hosts	373
8.5.1	Secure Shell Commands	374
8.5.2	Distributed Simulation and Visualization	375
8.5.3	Client/Server Programming	377
8.5.4	Threads	378
8.6	Classes	379
8.6.1	Class Programming	380
8.6.2	Checking the Class Type	383
8.6.3	Private Data	384
8.6.4	Static Data	385
8.6.5	Special Attributes	386
8.6.6	Special Methods	386
8.6.7	Multiple Inheritance	388
8.6.8	Using a Class as a C-like Structure	388
8.6.9	Attribute Access via String Names	389
8.6.10	New-Style Classes	390
8.6.11	Implementing Get/Set Functions via Properties	390
8.6.12	Subclassing Built-in Types	392
8.6.13	Building Class Interfaces at Run Time	394
8.6.14	Building Flexible Class Interfaces	398
8.6.15	Exercises	404
8.7	Scope of Variables	408
8.7.1	Global, Local, and Class Variables	408
8.7.2	Nested Functions	410
8.7.3	Dictionaries of Variables in Namespaces	411
8.8	Exceptions	413
8.8.1	Handling Exceptions	413
8.8.2	Raising Exceptions	415
8.9	Iterators	415
8.9.1	Constructing an Iterator	416
8.9.2	A Pointwise Grid Iterator	418
8.9.3	A Vectorized Grid Iterator	421
8.9.4	Generators	423
8.9.5	Some Aspects of Generic Programming	424
8.9.6	Exercises	428
8.10	Investigating Efficiency	429
8.10.1	CPU-Time Measurements	430
8.10.2	Profiling Python Scripts	433
8.10.3	Optimization of Python Code	434
8.10.4	Case Study on Numerical Efficiency	437

9	Fortran Programming with Numerical Python	
	Arrays	443
9.1	Problem Definition	443
9.2	Filling an Array in Fortran	445
9.2.1	The Fortran Subroutine	445
9.2.2	Building and Inspecting the Extension Module.....	446
9.3	Array Storage Issues	448
9.3.1	Generating an Erroneous Interface	448
9.3.2	Array Storage in C and Fortran	450
9.3.3	Input and Output Arrays as Function Arguments ...	451
9.3.4	F2PY Interface Files	458
9.3.5	Hiding Work Arrays	462
9.4	Increasing Callback Efficiency	463
9.4.1	Callbacks to Vectorized Python Functions	463
9.4.2	Avoiding Callbacks to Python	466
9.4.3	Compiled Inline Callback Functions	466
9.5	Summary	470
9.6	Exercises	470
10	C and C++ Programming with Numerical Python Arrays	475
10.1	C Programming with NumPy Arrays	476
10.1.1	The Basics of the NumPy C API.....	476
10.1.2	The Handwritten Extension Code	478
10.1.3	Sending Arguments from Python to C	479
10.1.4	Consistency Checks	480
10.1.5	Computing Array Values	480
10.1.6	Returning an Output Array	483
10.1.7	Convenient Macros	484
10.1.8	Module Initialization	485
10.1.9	Extension Module Template	486
10.1.10	Compiling, Linking, and Debugging the Module.....	488
10.1.11	Writing a Wrapper for a C Function	489
10.2	C++ Programming with NumPy Arrays	492
10.2.1	Wrapping a NumPy Array in a C++ Object	493
10.2.2	Using SCXX	495
10.2.3	NumPy-C++ Class Conversion	497
10.2.4	Migrating Loops to C++ with Weave.....	505
10.3	Comparison of the Implementations	506
10.3.1	Efficiency	506
10.3.2	Error Handling	509
10.3.3	Summary	510
10.4	Exercises	511

11 More Advanced GUI Programming	515
11.1 Adding Plot Areas in GUIs	515
11.1.1 The BLT Graph Widget	516
11.1.2 Animation of Functions in BLT Graph Widgets	522
11.1.3 Other Tools for Making GUIs with Plots	524
11.1.4 Exercises	525
11.2 Event Bindings	527
11.2.1 Binding Events to Functions with Arguments	528
11.2.2 A Text Widget with Tailored Keyboard Bindings	530
11.2.3 A Fancy List Widget	533
11.3 Animated Graphics with Canvas Widgets	536
11.3.1 The First Canvas Encounter	537
11.3.2 Coordinate Systems	538
11.3.3 The Mathematical Model Class	542
11.3.4 The Planet Class	543
11.3.5 Drawing and Moving Planets	545
11.3.6 Dragging Planets to New Positions	546
11.3.7 Using Pmw's Scrolled Canvas Widget	550
11.4 Simulation and Visualization Scripts	552
11.4.1 Restructuring the Script	553
11.4.2 Representing a Parameter by a Class	555
11.4.3 Improved Command-Line Script	569
11.4.4 Improved GUI Script	570
11.4.5 Improved CGI Script	571
11.4.6 Parameters with Physical Dimensions	572
11.4.7 Adding a Curve Plot Area	574
11.4.8 Automatic Generation of Scripts	576
11.4.9 Applications of the Tools	577
11.4.10 Allowing Physical Units in Input Files	582
11.4.11 Converting Input Files to GUIs	587
12 Tools and Examples	591
12.1 Running Series of Computer Experiments	591
12.1.1 Multiple Values of Input Parameters	592
12.1.2 Implementation Details	595
12.1.3 Further Applications	600
12.2 Tools for Representing Functions	604
12.2.1 Functions Defined by String Formulas	604
12.2.2 A Unified Interface to Functions	609
12.2.3 Interactive Drawing of Functions	615
12.2.4 A Notebook for Selecting Functions	619
12.3 Solving Partial Differential Equations	626
12.3.1 Numerical Methods for 1D Wave Equations	627
12.3.2 Implementations of 1D Wave Equations	630
12.3.3 Classes for Solving 1D Wave Equations	637
12.3.4 A Problem Solving Environment	643
12.3.5 Numerical Methods for 2D Wave Equations	649

12.3.6	Implementations of 2D Wave Equations	652
12.3.7	Exercises	661
A	Setting up the Required Software Environment . . .	663
A.1	Installation on Unix Systems	663
A.1.1	A Suggested Directory Structure	663
A.1.2	Setting Some Environment Variables	664
A.1.3	Installing Tcl/Tk and Additional Modules	665
A.1.4	Installing Python	666
A.1.5	Installing Python Modules	668
A.1.6	Installing Gnuplot	671
A.1.7	Installing SWIG	672
A.1.8	Summary of Environment Variables	672
A.1.9	Testing the Installation of Scripting Utilities	673
A.2	Installation on Windows Systems	673
B	Elements of Software Engineering	679
B.1	Building and Using Modules	679
B.1.1	Single-File Modules	679
B.1.2	Multi-File Modules	683
B.1.3	Debugging and Troubleshooting	684
B.2	Tools for Documenting Python Software	686
B.2.1	Doc Strings	686
B.2.2	Tools for Automatic Documentation	688
B.3	Coding Standards	692
B.3.1	Style Guide	692
B.3.2	Pythonic Programming	696
B.4	Verification of Scripts	701
B.4.1	Automating Regression Tests	701
B.4.2	Implementing a Tool for Regression Tests	705
B.4.3	Writing a Test Script	709
B.4.4	Verifying Output from Numerical Computations	710
B.4.5	Automatic Doc String Testing	714
B.4.6	Unit Testing	716
B.5	Version Control Management	718
B.6	Exercises	720
	Bibliography	725
	Index	727